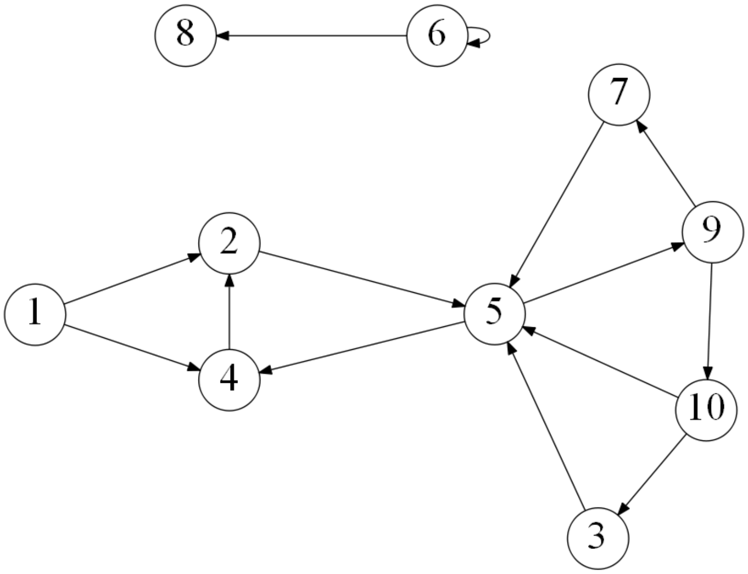
CS 385, Homework 3: Graph Algorithms

Name: Alex Johnson Date: 10/14/19

I pledge my honor that I have abided by the Stevens Honor System.

Point values are assigned for each question. Points earned: ____ / 100

Consider the following graph:



1. Draw how the graph would look if represented by an adjacency matrix. You may assume the indexes are from 1 through 10. Indicate 1 if there is an edge from vertex A -> vertex B, and 0 otherwise. (10 points)

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	1	0	0	0	0	0	0
2	1	0	0	1	1	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	1
4	1	1	0	0	1	0	0	0	0	0
5	0	1	1	1	0	0	1	0	1	1
6	0	0	0	0	0	1	0	1	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	1	0	1	0	0	1
0	0	1	0	0	1	0	0	0	1	0

2. Draw how the graph would look if represented by an adjacency list. You may assume the indexes are from 1 through 10. (10 points)

1	2	4
2	5	---------
3	5	---------
4	2	---------
5	4	---------
6	6	8

7	5	---------
8	---------	---------
9	7	10
10	3	5

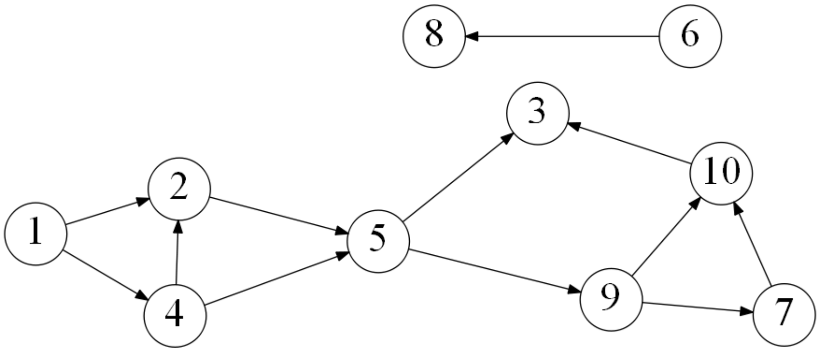
3. List the order in which the vertices are visited with a breadth-first search. If there are multiple vertices adjacent to a given vertex, visit the adjacent vertex with the lowest value first. (10 points)
   1,2,4,5,9,7,10,3,6,8
4. List the order in which the vertices are visited with a depth-first search. If there are multiple vertices adjacent to a given vertex, visit the adjacent vertex with the lowest value first. (10 points)
   1,2,5,9,7,4,10,3,6,8
5. a) What is the running time of breadth-first search with an adjacency matrix? (5 points)
   b) What is the running time of breadth-first search with an adjacency list? (5 points)
   a. $O(V^2)$
   b. $O(V+E)$

6. a) What is the running time of depth-first search with an adjacency matrix? (5 points)
   b) What is the running time of depth-first search with an adjacency list? (5 points)
   a. $O(V^2)$
   b. $O(V+E)$

7. While an adjacency matrix is typically easier to code than an adjacency list, it is not always a better solution. Explain when an adjacency list is a clear winner in the efficiency of your algorithm? (5 points)
   a. Adjencency lists are better when there is a low amount of edges and a high amount of vertexes. This is because there is less empty edges making the list take up less space. The list also uses $O(V+E)$ compared to $O(V^2)$ so if there is a smaller around of edges then vertices then V+E would be less then V^2.

8. Explain how one can use a breadth-first to determine if an undirected graph contains a cycle. (10 points)
   a. Use a BFS to go through the unvisited vertexes and mark them as visited. Then put the vertexes in a queue and remove if you don't get another visited. If you end with 0 in the queue then there is no cycle.
9. On undirected graphs, does either of the two traversals, DFS or BFS, always find a cycle faster than the other? If yes, indicate which of them is better and explain why it is the case; if not, draw two graphs supporting your answer and explain the graphs. (10 points)
   a. They would both have the same complexity and the DFS has $O(V+E)$. The DFS uses a stack however making it more efficient.
10. Explain why a topological sort is not possible on the graph at the very top of this document. (5 points)
    a. Since there is a loop on index 6 the topological sort will not work.

Consider the following graph:



11. List the order in which the vertices are visited with a topological sort. Break ties by visiting the vertex with the lowest value first. (10 points)
    a. 1,4,2,5,6,8,9,7,10,3